# Logical Discrete Event Systems in a trace theory based setting

Smedinga, R.

*Published in:*
EPRINTS-BOOK-TITLE



*Publication date:*
1993

# Logical Discrete Event Systems in a trace theory based setting

dr. R. Smedinga
department of computing science, University of Groningen
p.o.box 800, NL-9700 AV Groningen, the Netherlands
tel. +3150633937, email: rein@cs.rug.nl

June 4th, 1993

**Abstract**

Discrete event systems can be modelled using a triple consisting of some alphabet (representing the events that might occur), and two trace sets (sets of possible strings) denoting the possible behaviour and the completed tasks of the system. Using this definition we are able to formulate and solve control problems and define the notion of deadlock. Also distributed control will be discussed in this paper.

## 1 Observer view

Different approaches and notation exist in the field of logical discrete event systems. Here, we try to give the most simple and clear notation that is possible, without loosing expressive power. We use the trace theory, developed in 1984 [Sne85]. Trace theory was developed for VLSI-design, but seems also very suitable in modelling more general discrete event systems.

A logical discrete event system (LDES) is a system in which events happen. The only important aspects in the theory of logical discrete event systems are the order in which such events take place and (in our case) if some sequence of events completes some task.

A LDES can be seen as a black box. Suppose we have some observer who looks at that black box from a distance. He only has pen and paper and writes down a symbol on that paper each time he sees a corresponding event occurring in the system. At some time there will be a string of symbols on his paper. This sequence tells him what events have occurred and in what order. If the contents of the paper is $abc$ it means that first event $a$ occurred, then event $b$ and, last, event $c$. Systems do not necessarily behave in precisely one way, which means that the observer might also have seen some other events or order of events to happen. The system might also be in rest, i.e., do nothing at all, in which case the observer will have an empty piece of paper (which we denote by $\epsilon$).
We suppose that, in this view, it is also possible for the observer to see that some sequence of events is a completed task. A system has completed his task if it can legally stop its activities at that point. If this is the case, the observer writes on his paper a special mark to denote that the string he has written down so far denotes a completed task as well. If a system has completed a task it may finish, but (if more behaviour is possible) it may also decide to continue.

Now, we define *the* behaviour of a system as all possible sequences of events an observer of the system could have written down and *the* set of completed tasks as all possible sequences of events the observer could have written down marked as special.

Because the observer only has one pen he cannot write down two events at the same time. This means that two events cannot occur at the same time. If in some system two events (say $a$ and $b$) can occur in parallel, it is even likely that the observer writes down on his paper $a$ followed by $b$ as $b$ followed by $a$. If the behaviour of some system contains sequences $xaby$ and $xbay$ (where $x$ and $y$ are arbitrary sequences) we have no possibility to distinguish between parallel occurrence of $a$ and $b$ and choice between $ab$ and $ba$. This difference cannot be modelled in any language-based approach. Only Petri Nets have this possibility.

## 2 Definition

According to the intuitive ideas above we have the following definition of a LDES (see [Sme90, Sme92a, Sme92b, Sme92c]):

**Definition 1** *A logical discrete event system (LDES) is defined as a triple*

$$P = \langle \mathbf{a}P, \mathbf{b}P, \mathbf{t}P \rangle$$

*where:*
$\mathbf{a}P$ *is the alphabet (a finite set of symbols, representing the events),*
$\mathbf{b}P \subseteq (\mathbf{a}P)^*$ *the behaviour set (a, possibly infinite, set of strings over the alphabet),*
$\mathbf{t}P \subseteq (\mathbf{a}P)^*$ *the task set (also a, possibly infinite, set of strings over the alphabet).*

In order to obtain a *realistic system* we add the following restrictions to the definition:[1]

$$\mathbf{pref}(\mathbf{b}P) = \mathbf{b}P \qquad \mathbf{t}P \subseteq \mathbf{b}P$$

(i.e., the behaviour is prefix closed and each task is a behaviour). Unrealistic systems go beyond our intuitive idea of a LDES (how can an observer find a sequence $abc$ on his paper without ever having the sequence $ab$ on it). Nevertheless, unrealistic systems play a crucial role in finding a controller, as we shall see later.

## 3 Cooperation of systems

The next thing to define is the cooperation or connection of systems. Suppose we have two systems in connection. Suppose also that these systems have events in common. If an event is common to both systems it can only occur in one system if it can occur (at the very same time) in the other system as well. So a common event only occurs if both the systems can engage in it.

An other way of formalizing our intuitive idea of connecting systems is that each behaviour (and task set) in the connection must be such that a sequence, restricted to the alphabet of one of the systems, is an element of the behaviour (task set, respectively) of that system:

**Definition 2** *The connection of two systems $P$ and $R$ is defined by*

$$\begin{aligned} P \parallel R = \langle \mathbf{a}P \cup \mathbf{a}R, &\{x \mid x \in (\mathbf{a}P \cup \mathbf{a}R)^* \wedge x\lceil \mathbf{a}P \in \mathbf{b}P \wedge x\lceil \mathbf{a}R \in \mathbf{b}R\}, \\ &\{x \mid x \in (\mathbf{a}P \cup \mathbf{a}R)^* \wedge x\lceil \mathbf{a}P \in \mathbf{t}P \wedge x\lceil \mathbf{a}R \in \mathbf{t}R\} \rangle \end{aligned}$$

$\lceil$ stands for alphabet restriction: $x\lceil A$ is the projection of string $x$ on alphabet $A$, i.e., all symbols not in $A$ are removed from $x$.

It can easily be checked that the operator $\parallel$ is symmetric, idempotent, and associative and has a unit element $\langle \emptyset, \epsilon, \epsilon \rangle$ and a zero element $\langle \{\epsilon\}, \epsilon, \epsilon \rangle$

If we look at common events in a connection as internal events (not visible by the observer) the *external connection* can be used:

**Definition 3** *The external connection is defined by*[2]

$$P \rceil\lceil R = (P \parallel R)\lceil(\mathbf{a}P \div \mathbf{a}R)$$

The external connection is also symmetric, has the same unit element, but is not idempotent and only associative if each event only occurs in at most two of the systems.

[1] $\mathbf{pref}(X)$ denotes the set of all prefixes of strings of the set $X$.
[2] $P\lceil A = \langle \mathbf{a}P \cap A, \mathbf{b}P\lceil A, \mathbf{t}P\lceil A \rangle$ and $A \div B = (A \cup B) \setminus (A \cap B)$ is the symmetric set difference.

## 4 Ordering of systems

Systems can be ordered. We say system $P$ is a subsystem of system $R$, if everything that $P$ can do also can be done by $R$.

**Definition 4** *$P$ is a subsystem of $R$, notation $P \subseteq R$, if*

$$\mathbf{a}P = \mathbf{a}R \wedge \mathbf{b}P \subseteq \mathbf{b}R \wedge \mathbf{t}P \subseteq \mathbf{t}R$$

Notice that we only order systems with equal alphabets.

## 5 Control of events

Suppose system $P$ is given and the alphabet $\mathbf{a}P$ is divided into two parts:

$$\mathbf{e}P \cup \mathbf{c}P = \mathbf{a}P \qquad \mathbf{e}P \cap \mathbf{c}P = \emptyset$$

$\mathbf{c}P$ will contain all controllable events, i.e., events that might be common to other systems. $\mathbf{e}P$ contains the exogenous events. Those events should be controlled.

The general idea is that the events from $\mathbf{c}P$ are used to control the order of the events from $\mathbf{e}P$. So we are looking for a second system $R$, called a controller, with $\mathbf{a}R = \mathbf{c}P$. In order to specify the wanted order of the events from $\mathbf{e}P$ we define two more systems, $L_{min}$ and $L_{max}$ with

$$L_{min} \subseteq L_{max} \subseteq \langle \mathbf{e}P, (\mathbf{e}P)^* \rangle$$

**Control problem 1 (CODE)** *Given $P$, $\mathbf{e}P$, $\mathbf{c}P$, $L_{min}$, and $L_{max}$ as above, try to find $R$ with $\mathbf{a}R = \mathbf{c}P$ such that*

$$L_{min} \subseteq P \rceil\lceil R \subseteq L_{max}$$

The two systems $L_{min}$ and $L_{max}$ act as minimal and maximal needed exogenous behaviour of the system $P$. I.e., we want to restrict the uncontrolled exogenous system $P \lceil \mathbf{e}P$ to be within the given limits by using the controller $R$ and events $\mathbf{c}P$.

Finding such a controller can be done by using the following operator on systems, called the reflection operator:

**Definition 5** *The reflection of a system $P$, denoted $\sim P$, is defined by*

$$\langle \mathbf{a}P, (\mathbf{a}P)^* \setminus \mathbf{b}P, (\mathbf{a}P)^* \setminus \mathbf{t}P \rangle$$

Notice that if $P$ is realistic, $\sim P$ need not be. $\sim P$ may contain behaviour that is not prefix closed and/or tasks that are no behaviour.

## 6 Solution for code

We claim ( [Sme92a, Sme92c]) that

$$F(P, L) = \sim(P \rceil\lceil \sim L)$$

leads to a solution.

**Theorem 1** *The control problem has a solution if and only if*

$$L_{min} \subseteq P \rceil\lceil F(P, L_{max})$$

*and, if it is solvable, the greatest solution (with respect to $\subseteq$) is $F(P, L_{max})$.*

Because of using the reflection operator twice, the resulting controller need not be realistic any more. In order to find a realistic controller we must compute the so called des-interior of $F(P, L)$, where the des-interior of $P$ is defined by

$$\begin{aligned}\mathbf{des}(P) = \langle \mathbf{a}P, \{x \mid x \in \mathbf{b}P \wedge (\forall z : z \in \mathbf{pref}(x) : z \in \mathbf{b}P)\},\\ \{x \mid x \in \mathbf{t}P \wedge (\forall z : z \in \mathbf{pref}(x) : z \in \mathbf{b}P)\}\rangle\end{aligned}$$

# 7 Locked systems

Because we have defined a system using two sets of sequences of events, the behaviour and the task set, we are able to tell if at some point in behaviour a system is able to perform a completed task. If at some point no task can be completed any more, we say the system is *locked*. This (informal) definition of lock contains the well-known deadlock: not being able to do a next event (and not having completed a task), but also, what we will call, livelock: being able to do a next event at all times, but never being able to perform a task any more. Each system that contains deadlock cases, livelock cases, or combinations is called a system with the possibility of lock. The term deadlock appears in DES-literature more often as blocking.

**Definition 6** *We say a LDES has the possibility to lock if the set*

$$\mathbf{lock}(P) = \mathbf{b}P \setminus \mathbf{pref}(\mathbf{t}P)$$

*is not empty.*

Notice that $\mathbf{lock}(P)$ contains those behaviour that cannot be completed to become a task of $P$. Similar we say a connection of systems $P$ and $R$ may lock if $\mathbf{lock}(P \parallel R) \neq \emptyset$.

The controller $\mathbf{des}(F(P, L_{max}))$ may lead to a connection with $P$ that can lock. Therefore, we introduce a second control problem, similar to the previous one, but with an extra demand: the connection $P \parallel R$ of plant $P$ and controller $R$ should be free of lock. However, in order to find a solution for this problem, we first introduce some other problems:

**Control problem 2 (LFC)** *Given two systems $P$ and $R$ with $\mathbf{lock}(P \parallel R) \neq \emptyset$, find the greatest subsystem of $R$ that has a lock free connection with $P$.*

In order to solve this second problem we recall some results from [Sme90, Sme91a, Sme91b, Sme93]. Given some realistic systems $P$ and $R$ with $\mathbf{lock}(P \parallel R) \neq \emptyset$, we can construct a subsystem $R_{lf} \subseteq R$ such that $\mathbf{lock}(P \parallel R_{lf}) = \emptyset$. The following operator leads to this subsystem:

$$L(P, R) = R \setminus \mathbf{lock}(P \parallel R)\lceil \mathbf{a}R$$

where $R \setminus T$ is defined by

$$R \setminus T = \mathbf{des}(\langle \mathbf{a}R, \mathbf{b}R \setminus T, \mathbf{t}R \setminus T\rangle)$$

Although $T$ is only some trace set, we will denote the system $\langle \mathbf{a}P, T, T\rangle$ also by $T$. We have:

$$P \setminus T = P \cap {\sim}T = P \parallel {\sim}T$$

The greatest subsystem of $R$ that leads to a lock free connection with $P$ now is the greatest fix-point of $L(P, R)$, i.e., we iteratively compute:

$$\begin{aligned} R_0 &= R \\ R_{i+1} &= L(P, R_i) \end{aligned}$$

until that fix-point is reached. Let $\Lambda(P, R)$ denote that fix-point.

**Theorem 2** *$\Lambda(P, R)$ is the greatest subsystem of $R$ for which the connection with $P$ is free of lock.*

The above results can be used if $R$ is known. Sometimes we only have $P$ and want to control $P$ using $\mathbf{c}P$ in such a way that no lock can occur. This leads to the following additional problem:

**Control problem 3 (LF)** *Given some system $P$ with $\mathbf{lock}(P) \neq \emptyset$ and $\mathbf{c}P \subseteq \mathbf{a}P$ the set of control-events of $P$, find a controller $R$, with $\mathbf{a}R = \mathbf{c}P$, such that the connection $P \parallel R$ is free of lock and $R$ is minimal restrictive.*

The controller $R$ must be minimal restrictive, i.e., should be as general as possible. This suggests that we start with the most general controller that is possible: $R = \langle \mathbf{c}P, (\mathbf{c}P)^*, (\mathbf{c}P)^* \rangle$. Now applying the algorithm for solving the LFC-problem, starting with $R_0 = R$ we find a fix-point that turns out to be the greatest subsystem of $R$ that leads to a lock free connection with $P$ and, therefore, is minimal restrictive:

**Theorem 3** *If $\Lambda(P, \langle \mathbf{c}P, (\mathbf{c}P)^*, (\mathbf{c}P)^* \rangle) = \langle \mathbf{c}P, \emptyset, \emptyset \rangle$ no minimal restrictive controller for the LF-problem can be found, otherwise $\Lambda(P, \langle \mathbf{c}P, (\mathbf{c}P)^*, (\mathbf{c}P)^* \rangle)$ is the solution.*

## 8 Lock free control problem

The solutions of the previous control problems can now easily be combined to give a solution for the following control problem:

**Control problem 4 (LFCODE)** *Given a system $P$, control events $\mathbf{c}P \subseteq \mathbf{a}P$ and two systems $L_{min} \subseteq L_{max} \subseteq \langle \mathbf{e}P, (\mathbf{e}P)^*, (\mathbf{e}P)^* \rangle$ (with $\mathbf{e}P = \mathbf{a}P \setminus \mathbf{c}P$), find a controller $R$ with $\mathbf{a}R = \mathbf{c}P$ such that $L_{min} \subseteq P \rceil\lceil R \subseteq L_{max}$ and also $\mathbf{lock}(P \parallel R) = \emptyset$.*

This solution can now easily be found by combining the solutions of CODE and LFC: first we compute $R = \mathbf{des}(F(P, L_{max}))$ to find the greatest controller such that the connection $P \rceil\lceil R$ satisfies the minmax condition. Next, we compute the greatest subsystem of this $R$ that also leads to a lock free connection with $P$:

**Theorem 4** *A lock free controller for the control problem can be found if and only if*

$$L_{min} \subseteq P \rceil\lceil \Lambda(P, \mathbf{des}(F(P, L_{max})))$$

*The greatest solution then is given by $\Lambda(P, \mathbf{des}(F(P, L_{max})))$*

## 9 Distributed control

As a last part we will investigate distributed control, i.e., local control with global constraints. A number of possible configurations can be thought of:

- One global system with a number of local controllers
- Local subsystems working in cooperation with each other, where each local subsystem has to be controlled individually.

Having a global system with local control means having one system $P$ and (say 2) controllers $R_1$ and $R_2$ such that $P$ controlled by $R_1$ and $R_2$ has predefined desired exogenous behaviour. $R_i$ $(i = 1, 2)$ controls part of the events of $P$, namely $\mathbf{a}R_i$. The remaining events of $P$, i.e., $\mathbf{a}P \setminus (\mathbf{a}R_1 \cup \mathbf{a}R_2)$, are the exogenous events that should be controlled to behave according to some constraint. This leads to the following problem formulation:

**Control problem 5 (GsLc)** *The* Global System Local Control problem *is defined by:*
*Given a system $P$, alphabets $\mathbf{a}R_1$, $\mathbf{a}R_2$, and $\mathbf{e}P$ and global minimal and maximal constraints $L_{min}$ and $L_{max}$ with:*

$$\begin{array}{rr} \mathbf{a}R_1 \subseteq \mathbf{a}P & \mathbf{a}R_2 \subseteq \mathbf{a}P \qquad \mathbf{e}P \subseteq \mathbf{a}P \\ \mathbf{a}R_1 \cap \mathbf{a}R_2 = \emptyset & \mathbf{e}P = \mathbf{a}P \setminus (\mathbf{a}R_1 \cup \mathbf{a}R_2) \\ & L_{min} \subseteq L_{max} \subseteq (\mathbf{e}P)^* \end{array}$$

*find controllers $R_1$ and $R_2$ such that*

$$L_{min} \subseteq P \,]\![\, (R_1 \parallel R_2) \subseteq L_{max}$$

In the second case we have local (say 2) subsystems $P_1$ and $P_2$ working in cooperation. Each local system is controlled locally by a controller $R_i$ ($i = 1, 2$). The local control results in local exogenous behaviour $P_i \,]\![\, R_i$. The global exogenous behaviour should be according to some predefined minimal and maximal constraints. This leads to the following problem definition:

**Control problem 6 (LsLc)** *The* Local System Local Control problem *is defined by:*
*Given systems $P_1$ and $P_2$, alphabets $\mathbf{a}_{12}$, $\mathbf{c}P_1$, $\mathbf{c}P_2$, $\mathbf{e}P_1$ and $\mathbf{e}P_2$ and global minimal and maximal constraints $L_{min}$ and $L_{max}$ with:*

$$\begin{array}{rll} \mathbf{a}_{12} = \mathbf{a}P_1 \cap \mathbf{a}P_2 & & \textit{cooperating events} \\ \mathbf{c}P_i \subseteq \mathbf{a}P_i \qquad i = 1,2 & & \textit{control events} \\ \mathbf{a}_{12} \cap \mathbf{c}P_i = \emptyset \qquad i = 1,2 & & \textit{independency condition} \\ \mathbf{e}P_i = \mathbf{a}P_i \setminus \mathbf{c}P_i \qquad i = 1,2 & & \textit{exogenous events} \\ \mathbf{c}P_1 \cap \mathbf{e}P_1 = \emptyset \qquad \mathbf{c}P_2 \cap \mathbf{e}P_2 = \emptyset & & \mathbf{c}P_1 \cap \mathbf{c}P_2 = \emptyset \\ L_{min} \subseteq Lmax \subseteq (\mathbf{e}P_1 \cup \mathbf{e}P_2)^* & & \end{array}$$

*find controllers $R_1$ and $R_2$ with*

$$\mathbf{a}R_1 = \mathbf{c}P_1 \qquad \mathbf{a}R_2 = \mathbf{c}P_2$$

*such that*

$$L_{min} \subseteq (P_1 \,]\![\, R_1) \parallel (P_2 \,]\![\, R_2) \subseteq L_{max}$$

The general idea is that, having a system $P_1 \parallel P_2$ (i.e., built out of two other systems working in cooperation) find two separate controllers $R_1$ and $R_2$ working on $P_1$ and $P_2$ respectively, such that the total system acts as desired.

$P_1$ and $P_2$ cooperate through events $\mathbf{e}P_1 \cap \mathbf{e}P_2$. We call these events *cooperating events* in the distributed system and denote them by $\mathbf{a}_{12}$. The systems $P_1$ and $P_2$ can be seen as components of a system situated at different locations. The cooperation between these two processes is done via the cooperating events.

# 10 A solution for the LsLc-problem

The problem LsLc can be solved if we can find suitable $L_1^m$ and $L_2^m$ such that $P_1 \,]\![\, R_1 = L_1^m$ and $P_2 \,]\![\, R_2 = L_2^m$ and

$$L_{min} \subseteq L_1^m \parallel L_2^m \subseteq L_{max}$$

First, we introduce, for a given $P_1$, $P_2$, $L_{min}$, and $L_{max}$, some additional processes.

**Definition 7** *Associated with the LsLc-problem we introduce (for $i = 1, 2$):*

$$\mathbf{S}_i = L_{min} \lceil \mathbf{e}P_i \qquad i = 1,2$$

*called the* sureties,

$$\mathbf{V}_i = \sim \mathbf{S}_i \qquad i = 1, 2$$

*called the* vagues,

$$\mathbf{U}_i = ((\mathbf{V}_i \parallel \mathbf{S}_{i\oplus 1}) \setminus L_{max})\lceil \mathbf{e}P_i \qquad i = 1, 2$$

*called the* unusables ($\oplus$ *stands for addition modulo 2), and*

$$\mathbf{W}_i = \mathbf{V}_i \setminus \mathbf{U}_i \qquad i = 1, 2$$

*called the* warpers.

The sureties contain those traces that surely are needed to find $L_1^m$ and $L_2^m$. The vagues contain traces that may or may not be used to construct $L_1^m$ and $L_2^m$. Traces that cannot be used are in the unusables, traces that may be used are in the warpers.

**Lemma 1** *If* $L_{min}\lceil \mathbf{e}P_1 \parallel L_{min}\lceil \mathbf{e}P_2 \subseteq L_{max}$ *then the sureties and warpers associated with the LsLc-problem satisfy:*

$$L_{min} \subseteq \mathbf{S}_1 \parallel \mathbf{S}_2 \subseteq L_{max}$$
$$L_{min} \subseteq (\mathbf{S}_1 \cup \mathbf{W}_1) \parallel \mathbf{S}_2 \subseteq L_{max}$$
$$L_{min} \subseteq \mathbf{S}_1 \parallel (\mathbf{S}_2 \cup \mathbf{W}_2) \subseteq L_{max}$$

The main result for the LsLc-problem:

**Theorem 5** *Associated with the LsLc-problem we have:*

$$L_{min}\lceil \mathbf{e}P_1 \subseteq P_1 \rceil\lceil F(P_1, \mathbf{S}_1) \wedge L_{min}\lceil \mathbf{e}P_2 \subseteq P_2 \rceil\lceil F(P_2, \mathbf{S}_2)$$
$$\Rightarrow$$
*the LsLc-problem is solvable*

*and*

$$L_{min}\lceil \mathbf{e}P_1 \not\subseteq P_1 \rceil\lceil F(P_1, \mathbf{S}_1 \cup \mathbf{W}_1) \wedge L_{min}\lceil \mathbf{e}P_2 \not\subseteq P_2 \rceil\lceil F(P_2, \mathbf{S}_2 \cup \mathbf{W}_2)$$
$$\Rightarrow$$
*the LsLc-problem is not solvable*

If neither one of the conditions in this theorem is met, we cannot conclude solvability or unsolvability of the LsLc-problem. In that case, we have to do some tedious hand-work and try to find $W_1' \subseteq \mathbf{W}_1$ and $W_2' \subseteq \mathbf{W}_2$ with $W_1' \parallel W_2' = \langle \mathbf{e}P_1 \cup \mathbf{e}P_2, \emptyset, \emptyset \rangle$ and

$$L_{min}\lceil \mathbf{e}P_1 \subseteq P_1 \rceil\lceil F(P_1, \mathbf{S}_1 \cup W_1') \wedge L_{min}\lceil \mathbf{e}P_2 \subseteq P_2 \rceil\lceil F(P_2, \mathbf{S}_2 \cup W_2')$$

If we have found such $W_1'$ and $W_2'$, we have solutions for the LsLc-problem (namely $F(P_1, \mathbf{S}_1 \cup W_1')$ and $F(P_2, \mathbf{S}_2 \cup W_2')$). No conditions can be given yet when such $W_1'$ and $W_2'$ can be found.

## 11 Effectively computable

State graphs can be used to get an automaton-like representation of LDESs. It turns out that all operators used to compute solutions for our control problems can easily be translated into operators on these automatons, which enables us to compute the controllers effectively if the original systems we start with are regular (i.e., contain a finite number of states in there automaton representation). In [Sme92d] an example of the use of this theory can be found.

## 12 Conclusions

In this abstract we have shown how logical discrete event systems can be modelled using a simple approach. Moreover, using this modelling we are able to define and solve control problems and define and avoid lock. Some first results are given in the field of distributed control.

## References


[KBS92] P. Kozák, S. Balemi, and R. Smedinga, editors. *Discrete Event Systems: Modeling and Control, Proceedings of the joint workshop on Discrete event systems*, Progress in Systems and Control Theory, Prague, Czechoslovakia, August 26–28 1992. Birkhäuser Verlag, Basel, Switzerland.

[Sme90] R. Smedinga. Locked discrete event systems. Technical Report CS9002, Department of computing science, University of Groningen, 1990.

[Sme91a] R. Smedinga. Discrete event systems: deadlock, livelock, and livedeadlock. In U. Jaaksoo and V.I. Utkin, editors, *Automatic Control, world congress 1990, 13–17 August, proceedings of 11th IFAC world congress*, volume III, Tallinn, Estonia, USSR, 1991. Pergamon Press.

[Sme91b] R. Smedinga. An effective way to undo a discrete event system of its (dead)lock. In *Proceedings of the 1st IFAC Symposium on Design Methods of Control Systems*, Zürich, 4–6 September 1991.

[Sme92a] R. Smedinga. Discrete event systems. course-notes, Department of computing science, University of Groningen, 1992.

[Sme92b] R. Smedinga. An overview of results in discrete event systems using a trace theory based setting. In P. Kozák, S. Balemi, and R. Smedinga, editors, *Proceedings of the joint workshop on Discrete event systems*, Prague, Czechoslovakia, August 26–28 1992. Birkhäuser Boston, Inc.

[Sme92c] R. Smedinga. The reflection operator in discrete event systems. Technical Report CS9201, Department of computing science, University of Groningen, 1992.

[Sme92d] R. Smedinga. The workshop exercise using a trace theory based setting. In P. Kozák, S. Balemi, and R. Smedinga, editors, *Proceedings of the joint workshop on Discrete event systems*, Prague, Czechoslovakia, August 26–28 1992. Birkhäuser Boston, Inc.

[Sme93] R. Smedinga. Locked discrete event systems: how to model and how to unlock. *Journal on Discrete Event Dynamic Systems, theory and applications*, 2(3/4), 1993.

[Sne85] J.L.A. van de Snepscheut. *Trace theory and VLSI design*. Lecture notes in computer science, nr. 200. Springer Verlag, 1985.